Chapter 7

Computational approach in physics

7.1 Need for numerical analysis

Most of the problems in mechanics start with the equations of motion and analysis of the trajectory of motion. During the process of solving these problems with analytical methods, we are forced to neglect many parameters to avoid complexity. If the external parameters controlling the motion are large in number, the error is also large. As a case study, let us go through the one-dimensional motion.

Consider a freely falling body under gravity. Then,

$$m\frac{d^2x}{dt^2} = mg$$

$$\frac{d^2x}{dt^2} = g$$

For simplicity, we can assume that g is a constant. This can be used to calculate the velocity and position of the body at any time.

$$v = \frac{dx}{dt} = v_0 + gt$$

$$x_t = x_0 + v_0 t + \frac{1}{2}gt^2$$

But the solution based on the above equations are not agreeable with real values due to the following errors.

1. Gravity is not a constant in a wide range. It varies according to

   $$g' = \frac{g}{(1+{}^{h}/_{R})^2}$$

   Where h is the height from the surface of the earth and R is the radius of the earth.
2. Force of buoyancy due to the displaced water

   $$F = \frac{4}{3}\pi r^3 \rho g$$

   Where r is the radius of the body, ρ is the density of the medium, and g is the acceleration due to gravity.
3. Viscous force, $F=6\pi\eta r v$

   Where η is the coefficient of viscosity of the medium, r is the radius of the moving body and v is the velocity of the body
4. If C is the drag coefficient, ρ is the density of medium and r is the radius of the moving body.

   Air drag $F = \frac{1}{2}C\pi\rho r^2 v^2$
5. Effect of weather

A solution incorporating all these parameters will be highly complicated. In the numerical analysis, we can solve this problem step by step, including all the parameters, which we like to incorporate. By the fundamentals of differentiation

$$v_t = \lim_{h\to 0}\frac{x_{t+h} - x_t}{h}$$

$$a_t = \lim_{h\to 0}\frac{v_{t+h} - v_t}{h}$$

This is true only if $h\to 0$. That means h must be infinitesimally small. This can be made a reality only with a computer using any method like R-K methods, Monte-Carlo methods, Euler methods, etc. In the Euler method, when h becomes very small, the above set of equation become:

$$x_{t+h} = x_t + hv_t$$

$$v_{t+h} = v_t + ha_t$$

$$a = \frac{F(x,v,t)}{m}$$

Using these three equations we can estimate the acceleration, velocity, and displacement step by step using a computer simulation. Similarly, we can simulate all concepts in Physics:

7.2 Concept of Discretisation

The physical quantity which we are analyzing may be discrete or continuous. But the computer arithmetic is discrete and finite. So, to solve a problem using a computer, the variables must be discrete and finite.

For example, consider the problem of finding the velocity of a body after 3 seconds moving in a force field. Instead of solving it in a stretch, we will divide the time slot into several elements of minimum width, say Δt. Then from the initial value, calculate the velocity after Δt seconds. Then calculate the velocity in the next time element with this value. This can be continued to get the velocity at the end of 3 seconds. In this, we divided the total time span into a finite set of discrete-time elements as follows.

$t_1 = t_0 + h$

$t_2 = t_0 + 2h$

$t_3 = t_0 + 3h$

....

$T_n = t_0 + nh$

In general

$$t_{final} = t_{initial} + \sum_{k=1}^{n} \Delta t_k$$

Splitting of continuous quantity into small discrete elements is known as discretization. Using these discrete quantities, we can solve the problems of mechanics by many methods like Euler's method which we discussed in the last chapter.

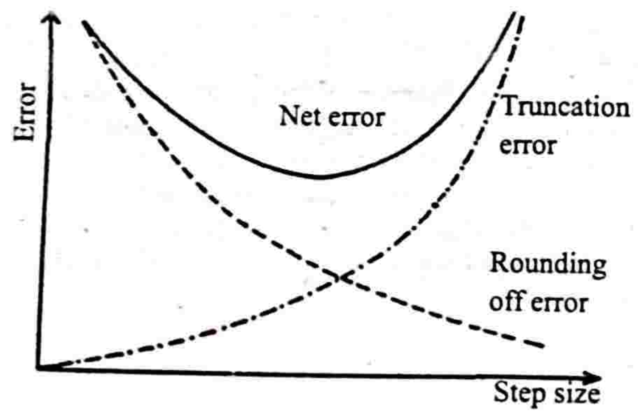
7.3 Selection of step size and accuracy

The accuracy of the numerical method is highly controlled by the step size. So, a judicious choice of step size is very important. There are two types of errors in the numerical analysis. They are **rounding off errors** and **truncation errors.**

7.3.1 Round off error

When we are assigning data to a memory location, the computer must convert it into binary format. So real numbers with a decimal point and fraction cannot be converted with full precession. Depends on the hardware of the computer and the selection of data type (Single precision or double precession), the system will round off the decimal part. This will make an error in the final result. Even though it is very small, it will sum up to a reasonable quantity in the end. This error is known as the **rounding off error**. When we opt for a small step size, the number of iterations increases. As a result, rounding off error also increases.

7.3.2 Truncation error



Truncation error is a characteristic of the algorithm. In the algorithm for the Euler method, **we assumed that the acceleration** and velocity are constant throughout in a time **element. Actually, it is not so.** For an easier flow of the algorithm, we partially omitted **the dependency of instantaneous** velocity on acceleration. This type of truncated dependencies will make some errors in the final result. This error is called **truncation error.** When step size decreases, truncation error decreases.

The net error in a programme is the sum of rounding off error and truncation error. But the efforts to decrease the truncation error will increase the rounding off error and vice versa. So, to reduce the net error, we need an optimum value of step size.

There is no straightforward rule for the selection of step size. Anyhow we can adopt the following steps for the proper selection of step size.

1. Select a reasonable step size to find the answer. Repeat it with gradually changed step sizes. At optimum step size, two consecutive final results will be almost the same. In other words, it will be the same for a particular number of decimal places. Then, that step size can be selected for accuracy up to that decimal place.
2. Select a reasonable step size. Check the final results with any one of conservation laws like conservation of energy, conservation of momentum, etc. For example, in the free fall problem, find the total kinetic energy and potential energy in different stages. If we are getting a constant for the total energy, our step size is optimum. Otherwise, repeat it with another step size.

7.4 Simulations based on Euler method

- To start simulation first we must study the exact nature of the system, dependencies, controlling equations, etc.
- Select the controlling equations or derive them. It may be polynomials or differential equations.
- Fid a numerical method of solution for these equations.
- Select one set of coordinates to start the graph. Depends on requirements, it may be a two-dimensional or three-dimensional system.
- Then find a set of initial values of the system (Boundary values).
- Using the Euler method, extend the coordinates to the required level. Care must be given to the selection of step-size which controls the accuracy.
- Plot it into a graph using Python matplotlib.
- For more, go through the following examples.

7.5 Position and velocity of a freely falling body-Euler method

By Euler method

$$a = \frac{dv}{dt}$$

$$a = \lim_{t \to 0} \frac{v - v_0}{t}$$

$$v = v_0 + at$$

If t is replaced by h (as a usual practice by Euler), $v = v_0 + ah$

$$v_{i+1} = v_i + ha_i$$

Similarly

$$v = \frac{dx}{dt}$$

$$v = \lim_{t \to 0} \frac{x - x_0}{t}$$

$$x = x_0 + vt$$

If t is replaced by h, $x = x_0 + vh$

$$x_{i+1} = x_i + hv_i$$

This is true for t → to 0. So divide the entire motion into a number of small elements(strips) depends upon the accuracy level. Then for each strip, the values can be calculated by the following four equations,

$$a = \frac{f}{m}$$

$$v_{i+1} = v_i + ha_i$$

$$x_{i+1} = x_i + hv_i$$

$$t = t+h$$

For a freely falling body, the acceleration at any instant is the acceleration due to gravity. If we are neglecting all controlling parameters like variation of acceleration due to gravity from place to place, force of buoyancy, viscous force, air drag, etc. acceleration at any instant can be treated as a constant. Then $a = 9.8\ m/s^2$

The given initial values of position and velocity will be the position and velocity in the first element. For a freely falling body, it can be zero. Using these values, the position and velocity of the next element can be calculated.

$$v_{i+1} = v_i + ha_i$$

$$x_{i+1} = x_i + hv_i$$

Repeat this up to the last element. Using these data set we can draw the graphs.

Python code for freely falling body-Tabular form - C.7.1

```
#free fall -Euler Method-Table
from numpy import *
tf=float(input('Enter the final time    '))
h=float(input('Enter the time step size  '))
n=round((tf/h))
a=9.8
v=zeros(n+1, dtype=float)
x=zeros(n+1, dtype=float)
t=zeros(n+1, dtype=float)
x[0]=input('Enter initial position  ')
v[0]=input('Enter initial velocity  ')
t[0]=0
print(' Time    Position    Velocity')
print(' %6.3f    %6.3f    %6.3f' %(t[0],x[0],v[0]))
for i in range(0,n):
    v[i+1]=v[i]+a*h
    x[i+1]=x[i]+v[i]*h
    t[i+1]=t[i]+h
    print(' %6.3f    %6.3f    %6.3f' %(t[i+1],x[i+1],v[i+1]))
#end
```

Output

```
Enter the final time          1
Enter the time step size      0.25
Enter initial position        0
Enter initial velocity        0
 Time    Position   Velocity
 0.000    0.000      0.000
 0.250    0.000      2.450
 0.500    0.613      4.900
 0.750    1.838      7.350
 1.000    3.675      9.800
```

Example 1:

A body is falling freely from a height under gravity. Find the velocity and position at the end of 1 second. Tabulate the values at an interval of 0.25 seconds.

Answer:

initial position = 0   Initial velocity = 0

Time = 1   Time step = 0.25

$v_{i+1} = v_i + ha_i$ and $x_{i+1} = x_i + hv_i$

Step 1 $v_0 = 0$ $x_0 = 0$ a = 9.8

Step 2 $v_{0.25} = 0 + 0.25 \times 9.8 = 2.45$

$x_{0.25} = 0 + 0.25 \times 0 = 0$

Step 3 $v_{0.5} = 2.45 + 0.25 \times 9.8 = 4.9$

$x_{0.5} = 0 + 0.25 \times 2.45 = 0.613$

Step 4 $v_{0.75} = 4.9 + 0.25 \times 9.8 = 7.350$

$x_{0.75} = 0.613 + 0.25 \times 4.9 = 1.838$

Step 5 $v_1 = 7.350 + 0.25 \times 9.8 = 9.800$

$x_1 = 1.838 + 0.25 \times 7.350 = 3.676$

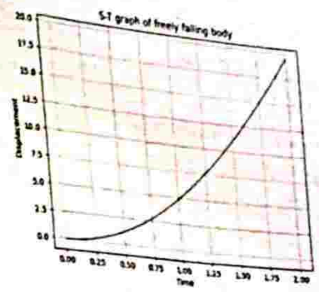
Time	Position	Velocity
0.000	0.000	0.000
0.250	0.000	2.450
0.500	0.613	4.900
0.750	1.838	7.350
1.000	3.675	9.800

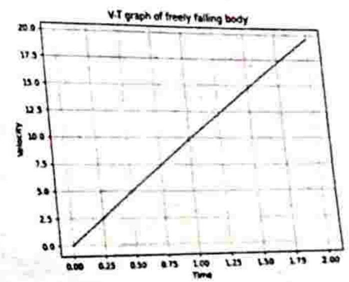
Python code for freely falling body-Graph - C.7.2

```
#freely falling body
from matplotlib.pyplot import *
from numpy import *
tf=float(input('Enter the final time     '))
h=float(input('Enter the time step size  '))
n=round((tf/h))
a=9.8
v=zeros(n+1, dtype=float)
x=zeros(n+1, dtype=float)
t=zeros(n+1, dtype=float)
```



```
x[0]=input('Enter initial position ')
v[0]=input('Enter initial velocity ')
t[0]=0
for i in range(0,n):
    v[i+1]=v[i]+a*h
    x[i+1]=x[i]+v[i]*h
    t[i+1]=t[i]+h
figure(1)
title('S-T graph of freely falling body')
xlabel('Time')
ylabel('Displacement')
grid(True)
plot(t,x)
figure(2)
title('V-T graph of freely falling body')
xlabel('Time')
ylabel('Velocity')
grid(True)
plot(t,v)
show()
```





Output

```
Enter the final time     2
Enter the time step size  0.01
Enter initial position  0
Enter initial velocity  0
```

7.6 Freely falling body in a viscous medium

For a freely falling body under gravity F=mg. According to the Stocks formula, the viscus force against gravity is F=6πηrv, Where η is the coefficient of viscosity of the medium, r is the radius of moving body and v is the velocity of the body.

Net downward force = mg-6πηrv. Net acceleration at an instant

$$a = g - \frac{6\pi\eta r}{m} v$$

$$= g - cv \quad \text{where } c = 6\pi\eta r/m$$

Let the initial value of a=0, v=0 and x=0 at t=0. From these values, by the Euler method, we can calculate the values of the next element by the equations,

$a_{i+1} = g + cv_i$

$v_{i+1} = v_i + ha_i$

$x_{i+1} = x_i + hv_i$

$t = t+h$

These can be repeated for the entire time. The result can be tabulated or visualise in a graph.

Python code for freely falling body in a viscous medium-Tabular form - C.7.3

```
#freely falling under viscocity
from numpy import *
tf=float(input('Enter the final time   '))
h=float(input('Enter the time step size  '))
n=round((tf/h))
a=zeros(n+1, dtype=float)
v=zeros(n+1, dtype=float)
x=zeros(n+1, dtype=float)
t=zeros(n+1, dtype=float)
vis=float(input ('Enter the Coefficient of viscosity  '))
rad=float(input ('Enter the radius of body  '))
mass=float(input ('Enter the mass of the body  '))
c=6*3.14*vis*rad/mass
a[0]=9.8-v[0]*c
x[0]=float(input('Enter initial position  '))
v[0]=float(input('Enter initial velocity  '))
t[0]=0
print(' Time    Position    Velocity    Acceleration ')
print(' %6.3f    %6.3f    %6.3f        %6.3f' %(t[0],x[0],v[0],a[0]))
for i in range(0,n):
  a[i+1]=9.8-v[i]*c
  v[i+1]=v[i]+a[i]*h
  x[i+1]=x[i]+v[i]*h
  t[i+1]=t[i]+h
```



```
  print('   %6.3f        %6.3f        %6.3f        %6.3f' %(t[i+1],x[i+1],v[i+1],a[i+1]))
#end
```

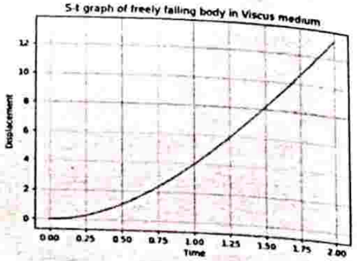
Output

```
Enter the final time     0.75
Enter the time step size  0.25
Enter the Coefficient of viscosity  0.7
Enter the radius of body  0.05
Enter the mass of the body  1
Enter initial position  0
Enter initial velocity  0
 Time     Position    Velocity   Acceleration
 0.000     0.000       0.000      9.800
 0.250     0.000       2.450      9.800
 0.500     0.613       4.900      8.184
 0.750     1.838       6.946      6.569
```

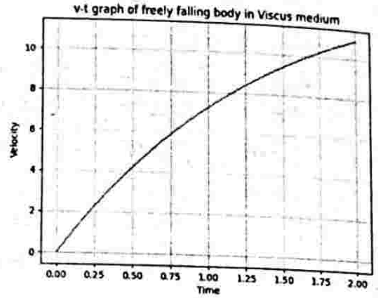
Python code for freely falling body in a viscous medium-Graph - C.7.4

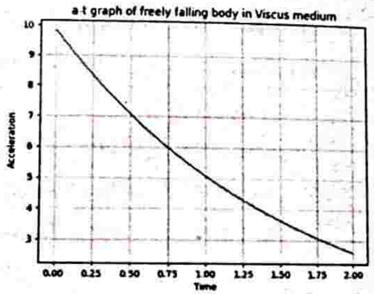
```
#freely falling under viscocity
from matplotlib.pyplot import *
from numpy import *
tf=float(input('Enter the final time   '))
h=float(input('Enter the time step size  '))
n=round((tf/h))
a=zeros(n+1, dtype=float)
v=zeros(n+1, dtype=float)
x=zeros(n+1, dtype=float)
t=zeros(n+1, dtype=float)
vis=float(input ('Enter the Coefficient of viscosity  '))
rad=float(input ('Enter the radius of body  '))
mass=float(input ('Enter the mass of the body  '))
c=6*3.14*vis*rad/mass
a[0]=9.8-v[0]*c
x[0]=float(input('Enter initial position  '))
```

```
v[0]=float(input('Enter initial velocity '))
t[0]=0
for i in range(0,n):
    a[i+1]=9.8-v[i]*c
    v[i+1]=v[i]+a[i]*h
    x[i+1]=x[i]+v[i]*h
    t[i+1]=t[i]+h
```




```
figure(1)
title('S-t graph of freely falling body in Viscus medium')
plot(t,x)
xlabel('Time')
ylabel('Displacement')
grid(True)
figure(2)
plot(t,v)
xlabel('Time')
ylabel('Velocity')
grid(True)
title('v-t graph of freely falling body in Viscus medium')
xlabel('Time')
ylabel('Velocity')
grid(True)
figure(3)
plot(t,a)
xlabel('Time')
ylabel('Acceleration')
grid(True)
title('a-t graph of freely falling body in Viscus medium')
show()
#end
```





Output

Enter the final time 2

Enter the time step size 0.01

Enter the Coefficient of viscosity 0.7

Enter the radius of body 0.05

Enter the mass of the body 1

Enter initial position 0

Enter initial velocity 0

Example 2:

A gently placed metallic ball of radius 0.05m is moving and mass 1 kg is moving down in castor oil of coefficient of viscosity 0.7PaS. Estimate the position and velocity after 0.75 seconds under the influence of viscous force. Use a step size of 0.25s.

Answer:

Initial position = 0 Initial velocity= 0

Time = 0.75 Coefficient of viscosity =0.7

Radius of body =0.05 Mass of the body = 1

Time step =0.25

$$c = \frac{6\pi\eta r}{m} = \frac{6 \times 3.14 \times 0.7 \times 0.5}{1} = 0.6594$$

$a_{i+1} = g - cv_i$, $v_{i+1} = v_i + ha_i$ and $x_{i+1} = x_i + hv_i$

Step 1 $a = 9.8\text{-}0 = 9.8$

$v_0 = 0$

$x_0 = 0$

Step 2 $a_{0.25} = 9.8 - 0.6594 \times 0 = 9.8$

$v_{0.25} = 0 + 0.25 \times 0 = 2.45$

$x_{0.25} = 0 + 0.25 \times 0 = 0$

Step 3 $a_{0.25} = 9.8 - 0.6594 \times 9.8 = 8.184$

$v_{0.5} = 2.45 + 0.25 \times 9.8 = 4.900$

$x_{0.5} = 0 + 0.25 \times 2.45 = 0.613$

Step 4 $a_{0.75} = 9.8 - 0.6594 \times 4.900 = 6.569$

$v_{0.75} = 4.900 + 0.25 \times 8.184 = 6.946$

$x_{0.75} = 0.613 + 0.25 \times 4.900 = 1.838$

Time	Position	Velocity	Acceleration
0.000	0.000	0.000	9.800
0.250	0.000	2.450	9.800

0.500	0.613	4.900	8.184
0.750	1.838	6.946	6.569

7.7 Two-dimensional motion (Projectile motion)

When we are making a two-dimensional analysis, the analysis methodology is the same as one-dimensional, but we must find acceleration, velocity, and displacement in x-direction and y-direction separately.

A projectile is a body projected upward with an initial velocity at an angle with a horizontal. In this case, initial velocity and acceleration in x-direction and y-direction are different. The time of flight is the same in x-direction and y-direction. Consider the influence of gravity only. Then

$F_x = 0 \quad F_y = -mg$

$a_x = 0 \quad a_y = -g$

Since the projectile is projected at an angle, The initial velocity can be split into two components. They are $v_0 \cos\theta$ along the x-axis and $v_0 \sin\theta$ along the y-axis. Then as discussed before, by the Euler method,

Along x direction

Acceleration $a_x = 0$

Initial velocity = $v_0 \cos\theta$

$$v_{x(i+1)} = v_{xi} + ha_x$$

$$x_{i+1} = x_i + hv_{xi}$$

Along y-direction

Acceleration = $-g$

Initial velocity = $v_0 \sin\theta$

$$v_{y(i+1)} = v_{yi} + ha_y$$

$$y_{i+1} = y_i + hv_{yi}$$

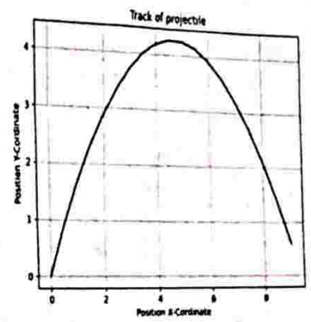
Using these formulas, the position and velocity at any stage can be calculated. In the case of a projectile, the maximum value of displacement in the x-direction is called the maximum range. The maximum value of displacement in the y-direction is called as the "maximum height".

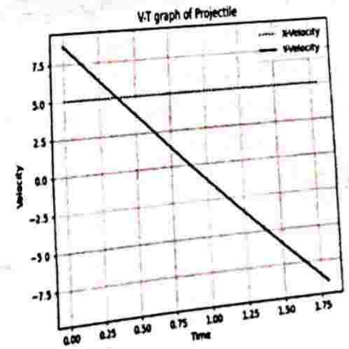
Python code for Projectile Motion-Graph - C.7.5

```
#Projectile-Euler Method-Graph
from matplotlib.pyplot import *
from numpy import *
h=float(input('Enter the time step size '))
```



```
n=round((tf/h))
t= zeros(n+1, dtype=float)
vx= zeros(n+1, dtype=float)
vy= zeros (n+1, dtype=float)
x= zeros (n+1, dtype=float)
y= zeros (n+1, dtype=float)
x[0]=float(input('Enter initial position-X Coordinate '))
y[0]=float(input('Enter initial position-Y Coordinate '))
v=float(input('Enter initial velocity '))
theta=float(input('Angle of projection in degree '))
theta=theta*pi/180
t[0]=0.0
vx[0]=v*cos(theta)
vy[0]=v*sin(theta)
ax=0.0
ay=-9.8
for i in range(0,n):
    vx[i+1]=vx[i]+ax*h
    vy[i+1]=vy[i]+ay*h
    x[i+1]=x[i]+vx[i]*h
    y[i+1]=y[i]+vy[i]*h
    t[i+1]=t[i]+h
figure(1)
title('Track of projectile')
xlabel('Position X-Coordinate')
ylabel('Position Y-Coordinate')
grid(True)
plot(x,y,color='k')
figure(2)
title('V-T graph of Projectile')
xlabel('Time')
ylabel('Velocity')
grid(True)
```





```
plot(t,vx,color='k',linestyle = ':')
plot(t,vy,color='k',linestyle = '-')
legend(['X-Velocity','Y-Velocity'])
show()
```

Output

```
Enter the final time      1.9
Enter the time step size  0.1
Enter initial position-X Coordinate  0
Enter initial position-Y Coordinate  0
Enter initial velocity  10
Angle of projection in degree   60
```

Python code for Projectile Motion-Tabular data - C.7.6

```
#Projectile-Euler method-Graph
from numpy import *
tf=float(input('Enter the final time     '))
h=float(input('Enter the time step size  '))
n=round((tf/h))
t= zeros (n+1, dtype=float)
vx= zeros (n+1, dtype=float)
vy= zeros (n+1, dtype=float)
x= zeros (n+1, dtype=float)
y= zeros (n+1, dtype=float)
x[0]=float(input('Enter initial position-X Coordinate  '))
y[0]=float(input('Enter initial position-Y Coordinate  '))
v=float(input('Enter initial velocity  '))
theta=float(input('Angle of projection in degree   '))
theta=theta*pi/180
t[0]=0.0
vx[0]=v*cos(theta)
vy[0]=v*sin(theta)
ax=0.0
ay=-9.8
print ('Time    X-Velocity    Y-Velocity    X-Position  Y-Position  ')
```



```
print ('%6.3f     %6.3f     %6.3f     %6.3f    %6.3f' %(t[0],vx[0],vy[0],x[0],y[0]))
for i in range(0,n):
    vx[i+1]=vx[i]+ax*h
    vy[i+1]=vy[i]+ay*h
    x[i+1]=x[i]+vx[i]*h
    y[i+1]=y[i]+vy[i]*h
    t[i+1]=t[i]+h
    print ('%6.3f    %6.3f     %6.3f     %6.3f    %6.3f' %(t[i+1],vx[i+1],vy[i+1],x[i+1],y[i+1]))

#end
```

Output

```
Enter the final time      0.6
Enter the time step size  0.2
Enter initial position-X Coordinate  0
Enter initial position-Y Coordinate  0
Enter initial velocity  10
Angle of projection in degree   60
Time    X-Velocity    Y-Velocity    X-Position  Y-Position
0.000     5.000         8.660         0.000       0.000
0.200     5.000         6.700         1.000       1.732
0.400     5.000         4.740         2.000       3.072
0.600     5.000         2.780         3.000       4.020
```

Example 6:

A body is projected with a velocity of 10 m/s at an angle 60^0. Tabulate the position and velocity for the first 0.6 seconds with an interval of 0.2 seconds.

Answer:

Initial velocity = 10m/s

Angle of projection = 60^0

Time = 0.6s

Time step =0.2s

Initial $v_x = 10 \times \cos 60 = 5.000$

Initial $v_y = 10 \times \sin 60 = 8.660$

$a_x = 0 \quad a_y = -9.8\ m/s^2$

$v_{x(i+1)} = v_{xi} + ha_x$

$v_{y(i+1)} = v_{yi} + ha_y$

$x_{i+1} = x_i + hv_{xi}$

$y_{i+1} = y_i + hv_{yi}$

Step 1: $v_{x0} = 5.000$

$v_{y0} = 8.660$

$x_0 = 0$

$y_0 = 0$

Step 2: $v_{x:0.2} = 5 + 0.2 \times 0 = 5.000$

$v_{y:0.2} = 8.66 - 0.2 \times 9.8 = 6.700$

$X_{0.2} = 0 + 0.2 \times 5 = 1.00$

$y_{0.2} = 0 + 0.2 \times 8.660 = 1.732$

Step 3: $v_{x:0.4} = 5 + 0.2 \times 0 = 5.000$

$v_{y:0.4} = 6.700 - 0.2 \times 9.8 = 4.74$

$X_{0.4} = 1 + 0.2 \times 5 = 2.00$

$y_{0.4} = 1.732 + 0.2 \times 6.700 = 3.072$

Step 4: $v_{x:0.6} = 5 + 0.2 \times 0 = 5.000$

$v_{y:0.6} = 4.74 - 0.2 \times 9.8 = 2.78$

$X_{0.6} = 2 + 0.2 \times 5 = 3.00$

$y_{0.6} = 3.072 + 0.2 \times 4.74 = 4.020$

Time	X-Velocity	Y-Velocity	X-Position	Y-Position
0.000	5.0	8.660	0.00	0.000
0.200	5.0	6.700	1.00	1.732
0.400	5.0	4.740	2.00	3.072
0.600	5.0	2.780	3.00	4.020

7.8 Radio Activate Decay-Euler Method

We are very familiar with the radioactive decay equation,

$$\frac{dN}{dt} = -\lambda N$$

It has a well known analytical solution to get the remaining number of nuclei after a definite time as,

$$N = N_0 e^{-\lambda t}$$

N is the number of remaining nuclei after t seconds, N_0 is the number of nuclei present in the sample at t=0 and λ is the decay constant.

But the radioactive decay has a better response with probability functions rather than the analytical method since the decay is a random process not controlled by external parameters. So, we can solve this problem in a better way using computational methods like Euler methods, R-K methods, Monte Carlo methods, etc. Now we can solve the problem with the Euler method.

$$\frac{dN}{dt} = \frac{N_{t+\Delta t} - N_t}{\Delta t}$$

$$N_{t+\Delta t} - N_t = \frac{dN}{dt} \times \Delta t$$

$$N_{t+\Delta t} = N_t + \frac{dN}{dt} \times \Delta t$$

Substituting for $\frac{dN}{dt}$, $N_{t+\Delta t} = N_t - \lambda N_t \times \Delta t$

$$N_{t+\Delta t} = N_t (1 - \lambda \Delta t)$$

Rewriting this equation as per the Euler method, replacing Δt with step size h

$$N_{i+1} = N_i(1 - \lambda h)$$

Python Code for Radio Activate Decay – Graph - C.7.7

```
#Programme for nuclear decay
from matplotlib.pyplot import *
from numpy import *
tf=float(input('Enter the final time    '))
h=float(input('Enter the time step size  '))
n=round(tf/h)
time=zeros(n+1, dtype=float)
nucl=zeros(n+1, dtype=float)
lam=float(input('Enter the disintegration Constant  '))
nucl[0]=int(input('Enter the initial number of nuclei '))
time[0]=0.0
for i in range(0,n):
    nucl[i+1]=nucl[i]*(1-h*lam)
    time[i+1]=time[i]+h
plot(time,nucl,color='k')
title('Nuclear Decay-Euler Method')
xlabel('Time')
```

```
ylabel('No of un-decayed nuclei')
grid(True)
show()
#end
```

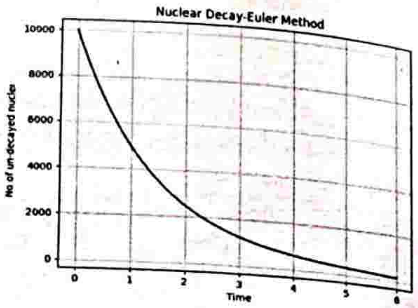
Output

Enter the final time 6

Enter the time step size 0.01

Enter the disintegration Constant 0.693

Enter the initial number of nuclei 10000



Python Code for Radio Activate Decay-number of Remaining Nuclei - C.7.8

```
#programme for nuclear decay
from numpy import *
tf=float(input('Enter the final time    '))
h=float(input('Enter the time step size  '))
n=round(tf/h)
time=zeros(n+1, dtype=float)
nucl=zeros(n+1, dtype=float)
lam=float(input('Enter the disintegration Constant  '))
nucl[0]=int(input('Enter the initial number of nuclei '))
time[0]=0.0
for i in range(0,n):
    nucl[i+1]=nucl[i]*(1-h*lam)
    time[i+1]=time[i]+h
print('After %6.3f second, No of remaining nuclei =
%d'%(time[i+1],nucl[i+1]))
#end
```

Output

Enter the final time 4

Enter the time step size 0.01

Enter the disintegration Constant 0.693

Enter the initial number of nuclei 10000

After 4.000 seconds, No of remaining nuclei = 619

Exercise

One word type questions

1. The angle of projection to get a maximum range for a projectile.
2. The angle of projection to get maximum value for the maximum height of a projectile.
3. When step size increases, rounding off error ---------
4. When step size increases, truncation error ---------
5. The element that controls the accuracy of a numerical method
6. ------ step size will minimize the error in a numerical calculation (Any, optimum, high, low)
7. In computational physics, a solution of two-dimensional motion is equivalent to ------ one-dimensional motion.

Short answer type questions

1. Suggest a method to find the maximum range of a projectile from the solution with the Euler method.
2. Suggest a method to find the maximum height of a projectile from the solution with the Euler method.
3. What is meant by air drag? How it can be calculated?
4. What is meant by the force of buoyancy? How it can be calculated?
5. What is meant by viscous force? How it can be calculated?
6. Explain the variation of acceleration due to gravity with height.
7. What is the rounding off error? How it can be reduced?
8. What is the truncation error? How it can be reduced?
9. Explain the effect of step size in the solutions with the Euler method.
10. Explain the need for optimisation of step size.
11. Explain the concept of discretisation.
12. Why the Euler method is considered as a better method than analytical method
13. What is meant by the range in the case of a projectile?
14. What is meant by the maximum height of a projectile?
15. Develop the necessary theory for the solution of radioactive decay by the Euler method.
16. When solving with a computer, suggest a logical method to find the terminal velocity of a freely following body due to air drag or viscus force.

17. What are the methods to make the proper selection of step size?
18. Explain the concept of error in the numerical analysis.
19. Why we are preferring zeros() to make a blank array rather than empty()
20. Explain the concept optimization of rounding off error and truncation error with the help of graph.

Problems/Programmes/Paragraph type questions

1. Write a python programme to tabulate the time, acceleration velocity, and position of a freely falling body considering the effect of viscosity.
2. Write a python programme to tabulate the time, acceleration velocity, and position of a freely falling body and to present a set of graphs considering the combined effect of gravity and viscous force.
3. Write a python programme to tabulate the time, acceleration velocity, and position of a freely falling body considering the force of buoyancy.
4. Write a python programme to find the terminal velocity of a freely falling body considering the air drag.
5. A body is falling under gravity against the flow of buoyancy. Estimate the velocity and position after 1 second. Do the calculations at an interval of 0.25 seconds. Given m=4 kg, r=1 cm, $\rho = 2400$ kg/s
6. A body is falling under gravity. Estimate the velocity and position after 6 seconds, considering the variation in the gravitational field. Do the calculations at an interval of 1 second. Given that the height of fall is 30 km, and the radius of the earth is 6400km.
7. A gently placed metallic ball of radius 0.2m and mass 1 kg is moving down in a liquid with the coefficient of viscosity 1PaS· Estimate the position and velocity after 1 second under the influence of viscous force. Use a step size of 0.25.
8. Make a tabulated chart of time, acceleration, velocity, and position of a freely falling body under gravity up to 1 second, by considering the opposing air drag. Solve with a time step of 0.25 seconds. Coefficient of drag = 0.45, Density of air 1.2 kg m^{-1}, Radius of body = 1m, Mass of body = 1kg.
9. Find the terminal velocity of a freely falling body considering the gravity which is a constant and the air drag only. Do the calculations by Euler equation with a time step of 0.2 seconds. Given that,

   Initial velocity = 0   Coefficient of air drag = 0.4
   Density of air = 1.2 kg m^{-1} Radius of body = 0.8m
   Mass of the body = 1kg

10. Write a program to find the vertical height from the ground at any instant of time for a body projected horizontally from a height considering the force of buoyancy.
11. Write a program to find the vertical height from the ground at any instant of time for a body projected horizontally from a height considering the air drag.
12. A body projected with a horizontal velocity of 1 m/s and moving under gravity. Make a tabulated chart of time, acceleration, velocity, and position up to 1 second, by considering the opposing air drag. Solve with a time step of 0.25 seconds. Coefficient of drag = 0.45, Density of air 1.2 kg m^{-1}, Radius of body = 1m, Mass of body = 1kg.
13. A metallic ball of radius 0.2m and mass 1 kg is projected on the surface of a liquid of coefficient of viscosity 1PaSwith a horizontal velocity of 1 m/s to move under gravity· Estimate the position and velocity after 1 second under the influence of viscous force. Use a step size of 0.25
14. A body is projected with a velocity of 9.8 m/s at an angle 40^0. Considering the effect of air drag, tabulate the position and velocity for the first 1.2 seconds by the Euler method with a time step of 0.4 seconds. Coefficient of drag = 0.45, Density of air 1.2 kg m^{-1}, Radius of body = 2m, Mass of body = 1kg.
15. Write a Python programme to track out the motion of a projectile considering the force of buoyancy opposing the motion
16. Write a Python programme to track out the motion of a projectile considering the variation of gravity and air drag.
17. Write a python programme to find the instantaneous velocity-position chart of an electron revolving around the nucleus in a hydrogen atom.
18. Develop a python programme for the simulation of radioactive decay by the Euler method.
19. Develop a python programme to find the number of remaining nuclei after a definite period by numerical method.

Long answer type questions

1. Explain the reasons by which we are recommending the numeric method over the analytical method for problems in Physics. Explain the Euler method to find the final position and velocity of a body after a particular time interval.
2. With the help of Python codes, explain the numerical method of tracking the motion of a projectile by the Euler method. Modify the method by introducing the concept of the force of buoyancy.
3. Explain the method of analysing the problems in mechanics by the Euler

method. Write the python code to simulate the position and velocity of a freely falling body under constant acceleration. Modify the method by introducing the concept of the force of viscosity.

4. Explain a method to simulate radioactive decay of a nucleus by the Euler method.

5. Find the number of remaining nuclei in a sample containing10000 nuclei of a radioactive material with disintegration constant 1.5, after 4 seconds. Take a step-size of 1 second.